**NEUROFLEET X**

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to define the **validation and testing strategy** for the NeuroFleetX platform. This plan will guide all testing activities, from initial unit tests to final user acceptance testing, to ensure that the system is **reliable, secure, and performs as expected**.

**1.2 Scope**

This document covers the testing of the entire NeuroFleetX system, including the **React frontend**, the **Node.js backend**, the **Firebase database**, and the integration with external services like the **Google Maps API**. It also covers the planned integration of the **Python AI microservice**.

**2. Testing Strategy**

We will employ a multi-layered testing approach, combining both **verification** (are we building the product right?) and **validation** (are we building the right product?).

**2.1 Types of Testing**

* **Unit Testing**: Individual components or functions (e.g., a single API endpoint, a React component) are tested in isolation. This will be performed by developers during the build process.
* **Integration Testing**: Tests the interaction between different modules or services (e.g., frontend sending a request to the backend, backend communicating with Firebase).
* **System Testing**: The entire, integrated system is tested to ensure all components work together as a single unit and meet the functional and non-functional requirements.
* **User Acceptance Testing (UAT)**: End-users (e.g., fleet managers, admins) will test the platform to validate that it meets their needs and functions correctly in a real-world scenario.
* **Performance Testing**: Simulates high loads to ensure the system remains stable and responsive. This is especially critical for the real-time tracking feature.
* **Security Testing**: Tests the system for vulnerabilities, including penetration testing and ensuring that the RBAC system functions correctly.

**2.2 Test Case Design Techniques**

* **Boundary Value Analysis**: Testing inputs at the extreme ends of their valid range (e.g., a user input field for age from 18-65 will be tested with 17, 18, 65, and 66).
* **Equivalence Partitioning**: Divides input data into partitions and tests one representative value from each partition (e.g., testing valid and invalid email formats).
* **State Transition Testing**: Used for testing user authentication, ensuring the system transitions correctly between states (e.g., 'logged out' -> 'logged in' -> 'logged out').

**3. Test Cases**

Here is a selection of test cases for key features. All test cases will follow a standard format: **ID**, **Description**, **Preconditions**, **Test Steps**, **Expected Result**.

| **Test Case ID** | **Feature** | **Description** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| **AUTH-01** | User Login | Verify a user can log in with valid credentials. | 1. Enter valid email and password. 2. Click login button. | The user is redirected to the dashboard, and a valid JWT token is received. |
| **AUTH-02** | RBAC Enforcement | Verify an 'Admin' can access the vehicle creation API, but a 'user' cannot. | 1. Log in as an 'Admin'. 2. Send a POST request to /api/vehicles. 3. Log in as a 'user'. 4. Send the same POST request to /api/vehicles. | The admin's request returns a 201 Created status. The user's request returns a 403 Forbidden status. |
| **RT-01** | Real-Time Tracking | Verify that a vehicle's location updates in real-time on the map. | 1. Navigate to the live tracking map. 2. Observe a vehicle's current location. 3. Broadcast a new simulated location from the backend. | The vehicle's marker on the map moves to the new location within 2 seconds. |
| **VM-01** | Vehicle CRUD (Admin) | Verify an admin can create a new vehicle. | 1. Log in as an 'Admin'. 2. Navigate to the vehicle management page. 3. Fill out the 'Add Vehicle' form with valid data. 4. Click 'Submit'. | A new vehicle is created and appears in the vehicle list on the dashboard. |
| **PT-01** | Predictive Maintenance (Planned) | Verify the AI model triggers a maintenance alert. | 1. Send simulated sensor data (e.g., extremely low tire pressure) to the AI microservice endpoint. 2. Check the system for a maintenance alert. | The AI microservice returns a 'high-risk' status, and a notification is triggered for the fleet manager. |
| **DPO-01** | Dynamic Routing (Planned) | Verify the system recommends an optimal route. | 1. Input a start and end destination. 2. Simulate traffic congestion on the fastest route. 3. Request a route recommendation. | The system provides an alternative route that avoids the traffic and shows the new estimated travel time. |

**4. Roles & Responsibilities**

* **Test Manager**: Oversees the entire testing process, manages the test team, and reports results to stakeholders.
* **Developers**: Perform unit and integration tests. Fix defects identified during all testing phases.
* **QA Team**: Designs and executes system, regression, and performance tests.
* **Project Manager**: Ensures resources are available and that testing aligns with project timelines.
* **End-Users**: Participate in User Acceptance Testing (UAT) to validate that the software meets their needs.